데이터통신 과제

— Router —

제 출 일	2016. 06. 12.
분 반	01분반
담당교수	김상하 교수님
학 과	컴퓨터공학과
학 번	201302450
조 장	석 지 훈
조 원	이 상 현
	임 대 영
	정 윤 수
	신 희 승

1. 실습개요

가. 실습 일시 및 장소

1) 실습 일시 : 2017. 06. 12. 10:00 ~ 24:00

2) 실습 장소 : 충남대학교 학생생활관 8동

나. 실습 목적

이번 실습의 목적은 직접 만든 정적 라우터를 이용하여 서로 다른 두 개의 네트워크에 있는 호스트와 통신을 성공하는 것을 목표로 한다. 저번 실습의 ARP 기능에다 IP Routing 기능을 추가하여 Routing Table의 Entry를 확인을 하여 서로 다른 네트워크 간의 통신을 가능하게 한다.

다. 실습 시나리오

1) Routing 프로그램 동작 전 수동으로 Topology에 맞게 routing table을 작성

2) ping 기능으로 다른 네트워크의 호스트와 통신 가능 여부를 확인.

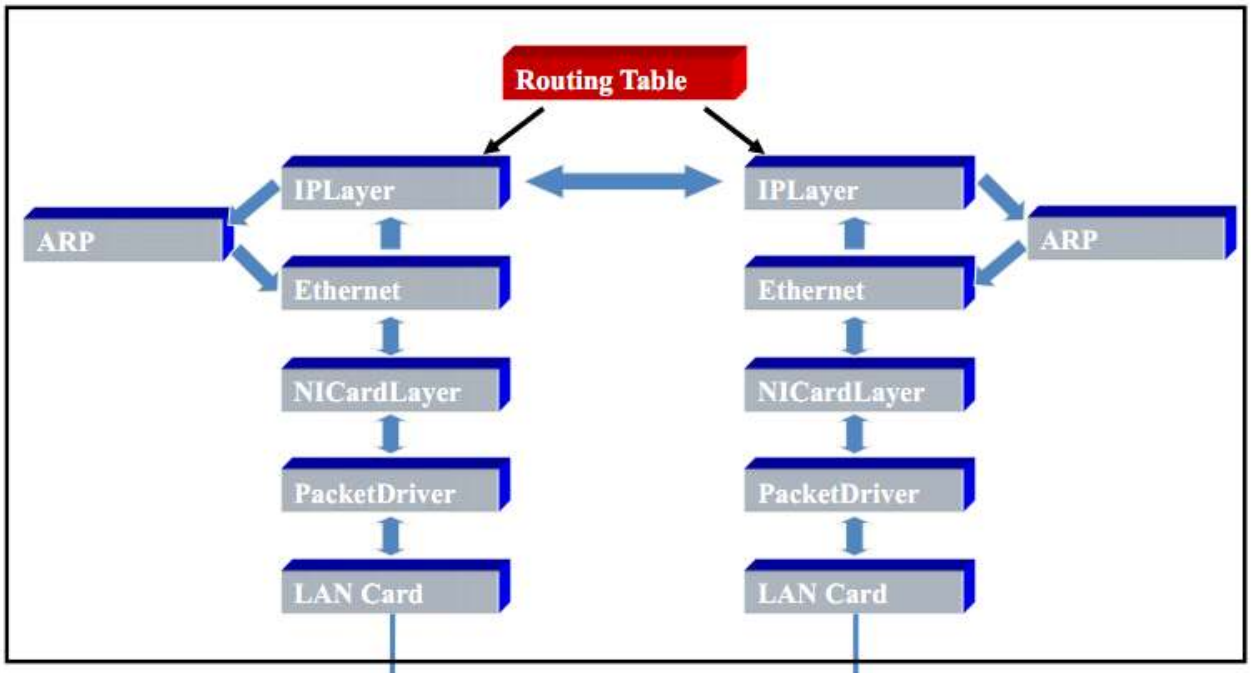
3) 호스트와 직접 연결이 된 라우터는 ping으로 보내진 packet을 수신하여 라우팅 테이블에서 확인, 매칭이되는 네트워크 인터페이스로 packet을 전달

4) 해당 subnet의 라우터에 도착 시 ARP cache table에서 해당 호스트에 대한 Ethernet address를 이용하여 packet 전달

5) 목적지 호스트에 도착하면 reply 메시지가 동일한 방식으로 출발, 호스트로 전달

2. 프로토콜 스택

가. 구조 설명(프로토콜의 역할 등)



1) StaticRouterDlg : UI를 담당하고 있다. 프로그램이 실행된 후 패킷이 오면 처리해주는 Thread와 Cache table Thread가 실행된다 . 그리고 각 버튼의 이벤트에 대한 함수를 가지고 있다.

2) IPLayer : 이 계층은 송신자가 입력한 데이터를 처리하는 과정을 진행하며 네트워크 계층이다. 기존과제와 달라진 점은 subnet mask와 연산을 통해 Routing table에 목적지 IP 주소를 찾는 과정이 추가된 것이다.

3) ARPLayer : 이 계층은 네트워크 계층 주소(IP)를 물리 주소로 변환하기 위해 사용된다. 목적지 호스트에 대한 MAC주소를 알아내기위해 ARP 질의 패킷을 구성하여 IP주소에 해당하는 물리주소 변환을 요청할 수 있다. 뿐만아니라 Proxy테이블을 이용해 라우터의 기능을 할 수 있고 MAC주소가 변환되었을 경우 새로 ARP 요청을 할 수 있다.

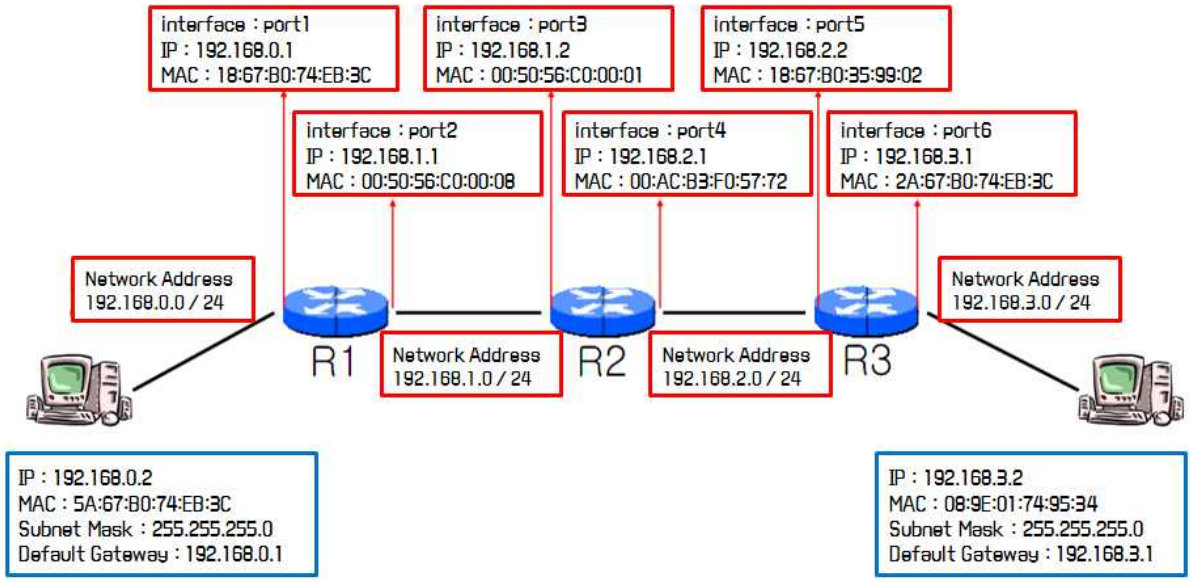
4) EthernetLayer : 이 계층은 호스트간에 통신을 담당하는 계층으로 데이터 링크 계층이다. 상위 Layer인 IP Layer에서 받아온 packet를 하위 Layer인 NILayer로 전송하고 하위 Layer로부터 온 데이터를 상위 Layer로 전송한다.

5) NILayer : 이 계층은 데이터를 패킷단위로 나누어서 네트워크로 연결된 다른 PC에 전송을 한다. 또한 수신에서는 연결된 네트워크로부터 받은 패킷들을 상위계층으로 보내준다.

3. 구현 설명

가. Topology

우리조의 Topology는 다음과 같다. R1, R2, R3는 랜카드가 2개 필요하므로 실습실의 컴퓨터를 사용한다. 사용되는 실습실의 컴퓨터에 따라 MAC주소가 바뀔 수 있으므로 데모연습시에 사용하였던 실습실 컴퓨터의 MAC주소를 기입하였고, 실제 데모에서는 데모에 사용되는 컴퓨터 interface의 MAC주소를 사용한다.



Router 1				
Destination	SubnetMask	Gateway	Interface	Flag
192.168.0.0	255.255.255.0	192.168.0.1	0	UH
192.168.1.0	255.255.255.0	192.168.1.1	1	U
192.168.2.0	255.255.255.0	192.168.1.2	1	UG
192.168.3.0	255.255.255.0	192.168.1.2	1	UG

Router 2				
Destination	SubnetMask	Gateway	Interface	Flag
192.168.0.0	255.255.255.0	192.168.1.1	0	UG
192.168.1.0	255.255.255.0	192.168.1.2	0	U
192.168.2.0	255.255.255.0	192.168.2.1	1	U
192.168.3.0	255.255.255.0	192.168.2.2	1	UG

Router 3				
Destination	SubnetMask	Gateway	Interface	Flag
192.168.0.0	255.255.255.0	192.168.2.1	0	UG
192.168.1.0	255.255.255.0	192.168.2.1	0	UG
192.168.2.0	255.255.255.0	192.168.2.2	0	U
192.168.3.0	255.255.255.0	192.168.3.1	1	UH

나. 라우터 프로그램 동작과정

```
void Rcv(TPARAM *param)
{
    TPARAM tparam = *param;
    *(param->flag1) = false;

    ALL_flag rcvflag;
    // Message
    UCHAR* buf[1514];

    UCHAR* ppayload;
    // 데이터 크기 지정
    ppayload = (UCHAR*)((long)buf + ETH_HSIZE + IP_HSIZE);
    while(tparam.dlg->isset){
        // Receive 수행 여부 판단
        if(!tparam.iplayer->Receive(ppayload, &rcvflag)){
            // APP 기능 확인
            if(rcvflag.app_type)
                ::PostMessage(HWND_BROADCAST, nRegUpdateData, NULL, NULL);
        }
    }
    *tparam.thrflag = false;
    return;
}
```

먼저 StaticRouterDlg클래스의 Receive Thread인 Rcv가 실행된다. 이 thread는 IPLayer의 receive를 시작으로 EthernetLayer의 receive, NILayer receive를 수행하고 pcap을 이용하여 패킷을 받아온다. 정상적으로 패킷을 받아왔다면 상위계층인 EthernetLayer에 패킷을 전달한다.

```
BOOL CEthernetLayer::Receive( unsigned char* ppayload, LPALL_flag flag)
{
    BOOL bSuccess = false ;
    ppayload = (UCHAR*)(unsigned long)ppayload - (ETH_HSIZE);
    if(mp_UnderLayer->Receive(ppayload, flag))
    {
        LPETHERNET pFrame = (LPETHERNET)ppayload;
        flag->app_type = ntohs(pFrame->enet_type); // 플래그에 상위의 어떤 프로토콜인지 지정
        if((addrcom(pFrame->enet_desaddr, broad) || addrcom(pFrame->enet_desaddr, ethsrc))
            && !addrcom(pFrame->enet_srcaddr, ethsrc))
            // 브로드 캐스트 거나 나한테 온것을 받아 들이나 내가 보낸것은 걸러냄
        {
            if(flag->app_type == ETHERTYPE_ARP)
            {
                return !m_ARP.getARPpacket(ppayload, flag);
            }
            else if(flag->app_type == ETHERTYPE_IP)
                return true;
        }
    }
    flag->app_type = 0;
    return false;
}
```

이후 EthernetLayer에서는 패킷의 EthernetLayer header에 적혀있는 목적지의 이더넷 주소를 확인하고 자신의 것이거나 브로드캐스트이면 상위계층으로 보낼 준비를 하며, 자신이 보낸 것이거나 자신의 것이 아니면 걸러낸다. 그리고 패킷에 적혀있는 app_type을 확인하여 ARP패킷인지 IP패킷인지 확인하고 ARP request패킷이면 ARPLayer인 ARPModule클래스의 getARPpacket을 실행한다.

```
if((Comip(pFrame->ip_desaddr, ipbroad) || Comip(pFrame->ip_desaddr, *ipsrc))
    && !Comip(pFrame->ip_srcaddr, *ipsrc)){
    // 선택한 이더넷에 대해 설정
    aitem.ethcard = flag->ethsel;
    aitem.success = true;
    aitem.regtime = time(NULL);
    aitem.ETHaddr = pFrame->enet_srcaddr;
    aitem.IPaddr = pFrame->ip_srcaddr;
    // 해당 TTL값을 20분으로 설정
    aitem.remindtime = 20 * 60;
    // 캐시 테이블에 입력
    m_cache.setARPCache(aitem);

    flag->app_len = ARP_HSIZE;
    flag->app_type = ETHERTYPE_ARP;
```

getARPpacket에서는 ARP cache table에 남아있을 수 있는 잔존시간을 20분으로 설정하고 arp cache table에 등록한다.

```
    // Request일 경우
    if(ntohs(pFrame->op) == ARP_OP_REQUEST){
        ALL_flag arpflag;
        arpflag.desip = pFrame->ip_srcaddr;
        arpflag.ethsel = flag->ethsel;

        // Reply의 실패에 따른 Error 메세지 설정
        if(!sendARP(&arpflag, ARP_OP_REPLY)){
            AfxMessageBox("ARP reply 실패");
        }
    }
```

그리고 패킷이 ARP request인 경우 sendARP메소드를 이용하여 ARP reply를 보내준다. 자세한 sendARP의 동작과정은 지난 과제의 연속이므로 구체적으로 설명하지 않는다.

```
BOOL CEthernetLayer::Receive( unsigned char* ppayload, LPALL_flag flag)
{
    BOOL bSuccess = false ;
    ppayload = (UCHAR*)(unsigned long)ppayload - (ETH_HSIZE);
    if(mp_UnderLayer->Receive(ppayload, flag))
    {
        LPETHERNET pFrame = (LPETHERNET)ppayload;
        flag->app_type = ntohs(pFrame->enet_type); // 플래그에 상위의 어떤 프로토콜인지 지정
        if((addrcom(pFrame->enet_desaddr, broad) || addrcom(pFrame->enet_desaddr, ethsrc))
            && !addrcom(pFrame->enet_srcaddr, ethsrc))
            // 브로드 캐스트 거나 나한테 온것을 받아 들이나 내가 보낸것은 걸러냄
        {
            if(flag->app_type == ETHERTYPE_ARP)
            {
                return !m_ARP.getARPpacket(ppayload, flag);
            }
            else if(flag->app_type == ETHERTYPE_IP)
                return true;
        }
    }
    flag->app_type = 0;
    return false;
}
```

만약 EthernetLayer에서 패킷에 IP라고 적혀있으면, 상위계층인 IPLayer로 패킷을 올려보낸다. 이는 호출스택을 따라 IPLayer의 Receive가 다음에 실행된다.

```
BOOL CIPLayer::Receive(unsigned char* ppayload , LPALL_flag flag){
    ppayload = (UCHAR*)(unsigned long)ppayload - IP_HSIZE;
    // 데이터를 수신할 경우
    if(mp_UnderLayer->Receive(ppayload, flag)){
        int i;
        ROUT_TABLE_ITEM ritem;
        // ARP의 응답일 경우
        if(flag->app_type == ETHERTYPE_ARP){
            LPARP pFramearp = (LPARP)ppayload;
            // 라우팅 테이블을 검색
            for(i = 0; i < mp_rtable->getEntryCnt(); i++){
                ritem = *mp_rtable->getEntry(i);
                flag->desip = pFramearp->ip_srcaddr;
                // 목적지를 넷 마스트와 비교
                if(Comip(ritem.des, Andip(ritem.netmask, pFramearp->ip_desaddr))){
                    return !mp_ARP->sendARP(flag, ARP_OP_REPLY, &pFramearp->ip_desaddr, &pFramearp->enet_srcaddr);
                }
            }
        }
```

IPLayer의 Receive에서는 하위계층에서 올라온 패킷을 이용한다. 만약 받은 패킷이 ARP reply인 경우는, 호스트A가 호스트B에 ARP request를 보내는 과정에서 자신을 거쳐서 보냈고, 호스트B가 그에 대한 ARP reply를 라우터에게 보낸 상황이다. 이 때 라우터는 받은 ARP reply패킷에서 목적지주소를 라우팅테이블 entry에 적힌 서브넷마스크와 마스킹을 하여 어느 곳으로 보내야 할지 결정한다. entry의 목적지주소와 일치하는 마스킹 값을 가지는 entry를 찾았다면, 해당 gateway주소로 ARP_reply를 보낸다. 이 때 패킷의 MAC주소는 라우터 자신의 것을 작성하여 보낸다.

```
flag->app_len = ntohs(pFrame->totlen);
// 자신이 보낸 패킷에 대한 return
if(Comip(pFrame->ip_desaddr, ipsrc)){ return false; }
// 라우팅 테이블 검색
for(i = 0; i < mp_rtable->getEntryCnt(); i++){
    ritem = *mp_rtable->getEntry(i);
    // 게이트웨이인 경우
    if(ritem.flag & GATE_FLAG){
        // AND 연산을 통한 주소확인
        if(Comip(ritem.des, Andip(ritem.netmask, pFrame->ip_desaddr))){
            SendParam param;
            ALL_flag sendflag = *flag;
            bool thrflag = true;
            // 목적 IP를 게이트웨이로 설정, x.x.x.0
            sendflag.desip = ritem.gateway;
            sendflag.ethsel = ritem.ethcard;
            param.flag = &sendflag;
            param.iplayer = mp_otherip[ritem.ethcard];
            param.ppayload = ppayload;
            param.thrflag = &thrflag;
            // send를 쓰레드를 구현하여 수행
            AfxBeginThread((AFX_THREADPROC)SendThr, &param);
            while(thrflag){
                Sleep(10);
            }
            return true;
        }
    }
```

만약 ARP reply패킷이 아니라면 해당 패킷을 라우팅테이블을 보고 forwarding해야 한다. 그래서 for문을 통해 라우팅테이블의 entry를 가져오고 entry에 적힌 flag를 보고 게이트웨이인 경우와 호스트인 경우를 분리하여 코드를 구현하였다.
플래그가 게이트웨이(G)인 경우 forwarding할 목적지주소를 라우팅테이블에서 찾은 entry의 게이트웨이주소로 설정하고 쓰레드를 통해 패킷을 보낸다.

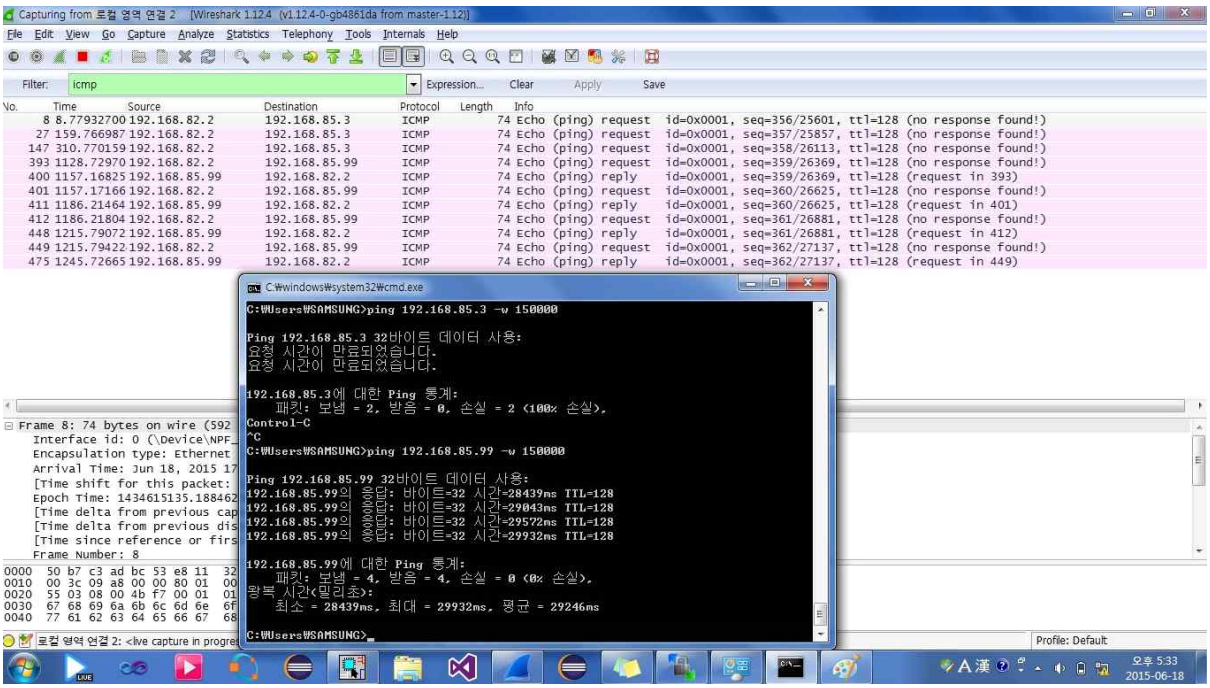
```
else if(ritem.flag & HOST_FLAG){
    if(Comip(ritem.des, Andip(ritem.netmask, pFrame->ip_desaddr)))
    {
        SendParam param;
        ALL_flag sendflag = *flag;
        bool thrflag = true;
        sendflag.desip = pFrame->ip_desaddr;
        sendflag.ethsel = ritem.ethcard;
        param.flag = &sendflag;
        param.iplayer = mp_otherip[ritem.ethcard];
        param.ppayload = ppayload;
        param.thrflag = &thrflag;
        AfxBeginThread((AFX_THREADPROC)SendThr, &param);
        while(thrflag){
            Sleep(10);
        }
        return true;
    }
}
```

플래그가 호스트(H)인 경우 forwarding할 목적지주소를 전달하고 있는 패킷의 IP header에 적혀있는 실제 전달되야하는 목적지주소로 설정하고 쓰레드를 통해 패킷을 보낸다.

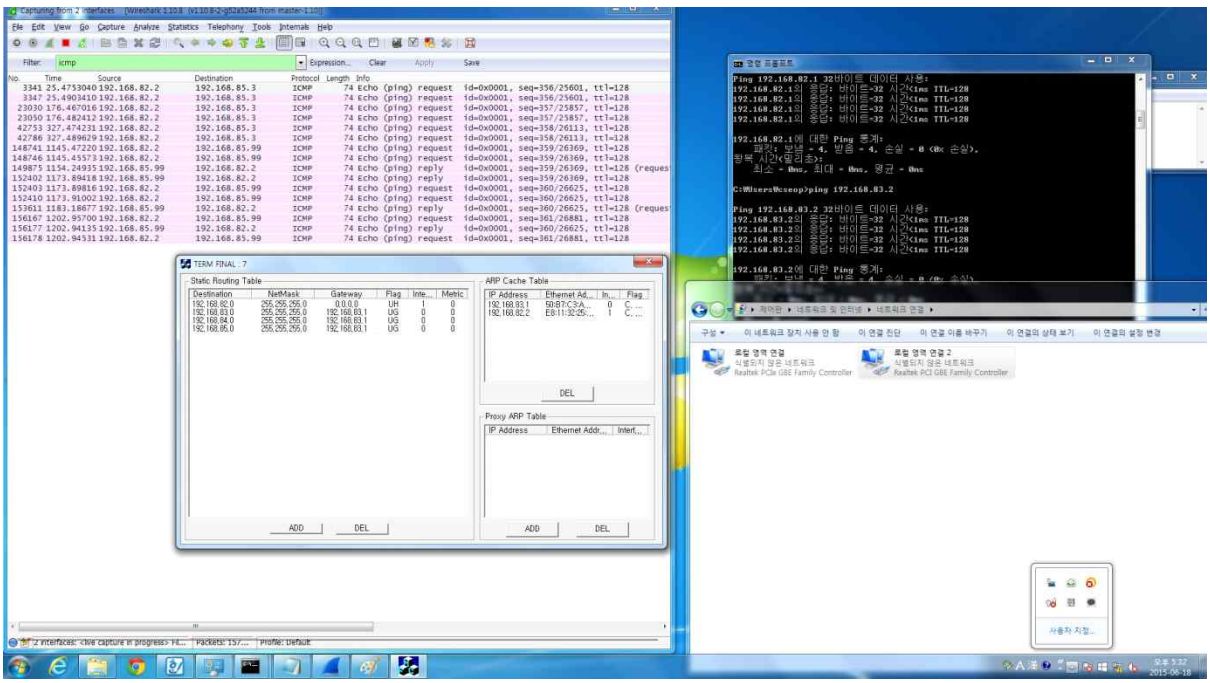
4. 실행 결과

가. 프로그램 결과 화면

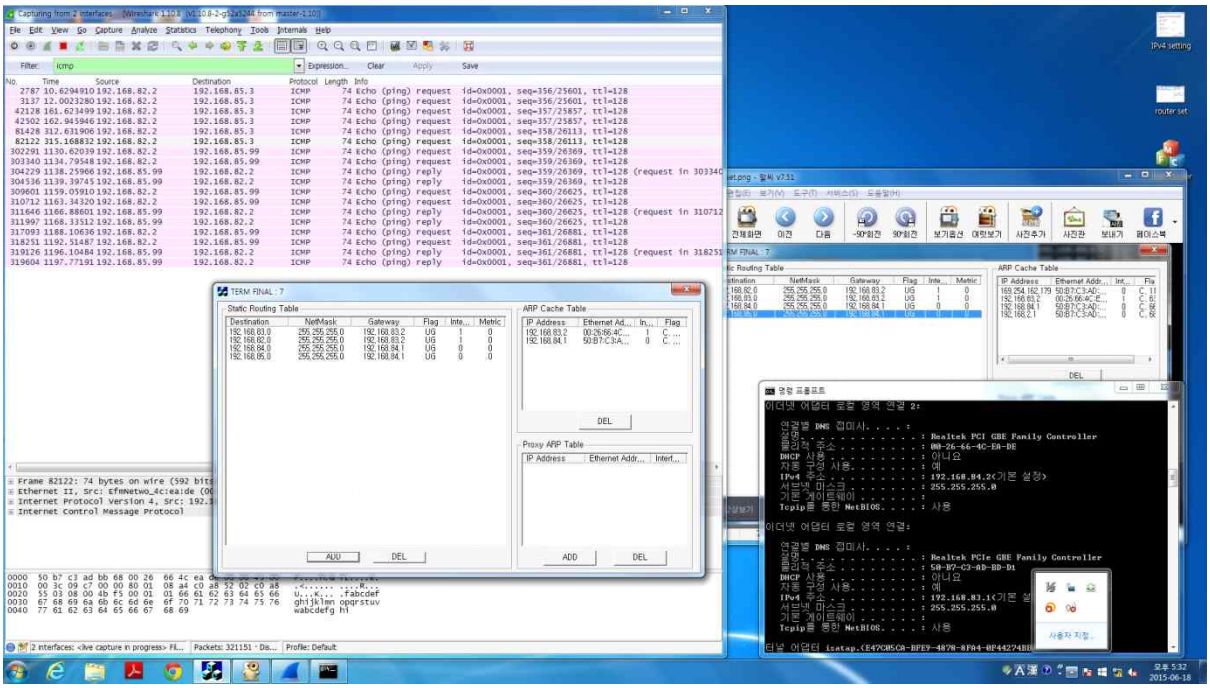
1) host1 Routing table 및 ping request&reply



2) Router1 Routing table 및 ping request&reply, ARP



3) Router2 Routing table 및 ping request&reply, ARP



4) Router3 Routing table 및 ping request&reply, ARP